Internetware Structure Description and Research of the Petri Net Method

Zhijian Wang
Information Science School, Guangdong University of Business Studies, Guanzhou, P.R.China
Email: zjian@gdcc.edu.cn

Yuping Hu[1 2]
[1] Key Lab of E-Business and Information Security, Hangzhou Normal University, Hangzhou, P.R.China
[2]Guangdong Key Lab of Electronic Commerce, Guangdong University of Business Studies, Guanzhou, P.R.China
Email: okhyp@yahoo.com.cn

Shaohua Li
Information Science School, Guangdong University of Business Studies, Guanzhou, P.R.China
Email: sohually@163.com

Dingguo Wei
Information Science School, Guangdong University of Business Studies, Guanzhou, P.R.China
Email: weidg@scnu.edu.cn

***Abstract*—Architecture design and specification is more important than algorithm or data structure establishment for a complex system, existing Architecture Description Languages are difficult to support the complete Internetware developing process. Petri nets describe system graphically and mathematically, provide a way to solve the problems in Internetware description and development. In this paper we investigate Internetware architecture description methods, current application of Petri nets in Internetware and the deficiency; present basic principles to describe Internetware architecture using Petri nets, the possible architecture model and architecture evolution mechanisms; discuss subnet based Web services composition description technologies; present the idea of developing Internetware using Petri nets as a whole process unified model. The Petri net method helps to transfer among different stages of Internetware developing easily; the smooth transition from architecture description to code implementation makes sure a good design can deduce a good realization.**

***Index Terms*—Architecture Description Language (ADL); Software architecture; Unified Modeling Language (UML); Internetware; Petri nets**

I. INTRODUCTION

Software systems were used to based on centralized and close computing platforms, but now they are in the passage from those traditional platforms to the Internet based open ones[1]. From the viewpoint of technology, software entities based on technologies including software component distribute among different nodes in Internet in an open and independent way, under an open environment a software entity can be released in an appropriate fashion, it can connect and cooperate with other entities crossing different networks by using corresponding protocols, these software entities form Internetwares.

Traditional software developing methods aim at steady structure systems, base on close and static platform. The process of building such a system is orderly, compose of system analysis and designing step by step according to the aimed question, and thus the basic functions and structure of the system are decided gradually. The internet is open, alterable and dynamic, but a traditional software developing method take none of these Internet prosperities into account, so it is difficult to satisfy the requirements of an Internetware, in which dynamic, cooperation, accommodation and evolvement are greatly desirable.

Component based software reuse is the necessary technology of modern software engineering. Because the network environment is open and dynamic, and requirements of different users are individual, the Internetwares are different from those traditional ones; an Internetware should be able to apperceive the dynamic change of surrounding environment and evolve accordingly, by so that it can try its best to satisfy the users with suitable functions, best performance and creditability. By following certain cooperating protocols, an Internetware can meet a group of different but consistent targets in a dynamic environment. The property of polymorphism helps Internetwares to provide flexibility and individuation for different users.

Petri nets describe system behaviors not only graphically but also mathematically, and thus reflect properties such as parallel, synchronization and resource sharing in a simple and intuitionistic way, and provide powerful mathematical analyzing ability at the same time.

doi:10.4304/jsw.6.9.1779-1786

Petri nets are widely used in system simulation, workflow modeling and many other fields, as an outstanding modeling method, Petri nets provide a way to solve the problems we met in Internetware description and development.

The paper is arranged in the following way: section 2 investigates software architecture description methods and current research about Internetware; section 3 analyzes current application of Petri nets in Internetware and the main deficiency, and the possible way to solve these questions; section 4 discusses the basic principles to describe Internetware architecture, possible Internetware architecture model, architecture evolution mechanisms and how to design the reduction rules set; section 5 discusses subnet based Web service composition description technologies, including how to ensue subnet's running ability, how to analyze and control component complexity; the possibility to develop Internetware using Petri nets as a whole process unified model is investigated in section 6 and a few conclusions are given in section 7.

II. ARCHITECTURE DECSCRIBTION OF INTERNETWARES

Current research about Internetware focus on following key questions: software model, architecture and corresponding theory, software reliability and creditability, quality of service and corresponding evaluating method, software developing method and so on. System architecture describes a kind of systems universally, so the realization of a special system is in fact the process of instantiating the given architecture. In the age of structural programming, usually a system is with limited scale, so software architecture is not specially given attention to, because a favorable structure is relatively easy to be obtained by structural designing, concretely we usually use a top down method and pay attention to the coupling among modules. But for those large scale and complex systems, when all comes to all, architecture design and specification is much more important than the establishment of an algorithm or data structures. Researching software architecture roundly and in depth is the most efficient way to improve software productivity and to maintain system.

Software architecture derives from software engineering, when people begin to research software architecture, basic ideas such as layered structure and other technologies from computer architecture and network architecture are used for reference. In recent years the research of software architecture becomes independent to that of software engineering and forms a new study branch. Main contents in software architecture research involve architecture description, style, evaluation and formalization. Reuse, quality and maintenance are the key factors to research software architecture.

Software architecture description methods today are divided into two types: formalization and visualization. The formal descriptions use certain Architecture Description Language (ADL), more than 20 ADLs have been defined by researcher from different countries and new ADLs continue to come up[1]. An ADL usually bases on certain formalization theory such as CSP and sequential logic, so it is born with strict syntax and semantics that can support system description, analysis, refinement and validation effectively. Because ADLs are not intuitive, they are not widely used today. Visual descriptions use traditional diagram composed by panes and lines, they are intuitive but they can not describe dynamic demands in an Internetware. Unified Modeling Languages (UML) are designed with rich semantic, universal, easy-to-understand and communicate, they are the de facto industry standard for visual modeling languages and are the most important visualization methods. UMLs provides rich view from multiple perspectives to describe a system, they can be effectively used in software system modeling, analysis and design.

Booch gave out a UML based model description including design view, procedure view, implementation view, deployment views and user case view [2]. However, UMLs are not good at describing elements such as connectors in software architecture, they describe the architecture in a non-formal or semi-formal way[2].

Generally speaking, current ADLs support description and analysis of either senior level abstraction of system architecture, or a particular architectural style as C2[3], usually different languages are used in criterion specification and implementation, these characteristics make it difficult to support architecture based software developing effectively, for example by using UMLs[4].

Service Oriented Architecture (SOA) is researched in order to combine different independent elementary services into flexible and complex service, that's Web Services Composition (WSC), in this way software can be reused. 3 requirements for description model of Web service combination are presented in reference [5]:

(1)The model owns adequate expression ability;

(2)The model can be transformed to an operational model directly;

(3)The model does not depend on a specific implementation language.

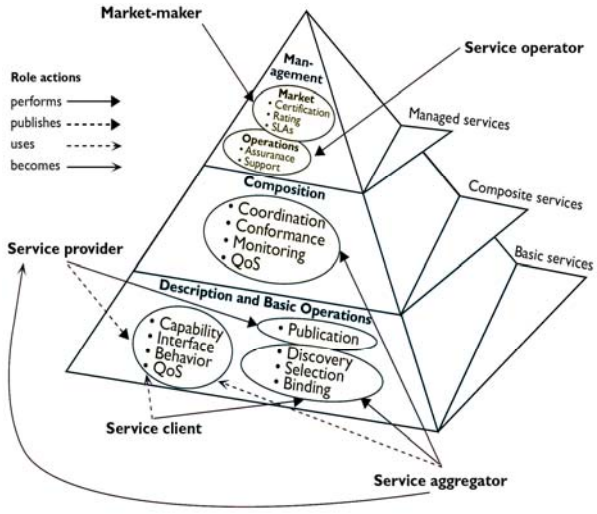

Figure 1. Extended service-oriented architecture:

The extended SOA model (figure 1) in reference [6] introduces a service composition layer in order to improve WSC needs. Existing ADLs for WSC include: Web Services Description Language (WSDL), Web Services Flow Language (WSFL), Business Process Execution Language for Web Services (BPEL4WS), OWL-S (Ontology Language for Web Services), Web Service Choreography Interface (WSCI) and so on, these ADLs describe component features, functionalities and their aggregation statically, they are unable to implement system dynamic description and system verification [5] .

Kramer and Magee have performed a series of studies in software architecture, they attempted to describe system architectures clearly using traditional static ADLs, and they did finish some related application in this way. For dynamic applications similar to Web based service, they present the concept of customizable management. They defined a three-layer architecture, take the turns of components controlling layer, changes managing layer and goals managing layer from the bottom to the top. The components controlling layer consists of a series of components to achieve corresponding tasks; the changes managing layer adjusts relationships between underlying components quickly according to some preplanned arrangements, so as to match the changes from corresponding underlying components or from upper layer targets [7]. In Kramer's model the key question of components controlling is to maintain the model's reliability and creditability in the process of adjusting system dynamically, this is quite different from a static system whose structural is stable and so its safety and reliability are guaranteed during the system design process.

Reference [8] researched identity, capability, behavior and other problem about system trustworthiness in an Internet based virtual computing environment (iVCE), proposed three mechanisms including inter domain authorization management, reliable service delivery and autonomous system collaboration, in order to provide users with consistent, reliable and transparent services. A system is represented as binary groups <AE.VEU>, where AE is the set of autonomous elements, VEU is the set of virtual execution units, and an execution is in fact the instantiation of an autonomous element which is assigned different parameters. Trustworthiness of an element in AE is guaranteed by its definition which is a triple group<i,c,b>. The structure of an IVEC system is a three-layer model, bottom-up are the virtual resource layer, the service layer and the user layer, the basic idea is close to that of Kramer's Model, however the middle layer is described in more detail and the method is also slightly different. Reference [9] defined an architecture ComDeValCo including functions of component definition and verifying, reference [10] gave a two-layer model composed of the target layer and the control layer.

III. PETRI NET BASED ADL

To describe Internetware architecture using Petri nets, capability and support like that of an ADL in software architecture are required. To consider in a relative large scope, the following functions should be provided: software architecture description, design process support, static and dynamic analysis capability, to support the evolution of software architecture, support for software architecture refinement, software architecture simulation and running; architecture description, service discovery. Structure evolution and component-based refinement are problems to be solved in the first step.

Some researchers have begun to apply Petri nets in Web services composition. Aalst presented WF_net, a kind of SISO Petri net model for workflow modeling [11], a WF_net needs to meet three basic requirements, and those who meet these three conditions are said to be soundness. WF_net has a large impact in workflow researching field, and also has some applications in the study of WSC nowadays [11, 12].

On the basis of WF-net, Hamadi and Benatallah proposed 8 calculation rules for WSC [13]. Zhovtobryukh improved Hamadi and Benatallah's result, he classified the calculation rules into seven types, that's order, exclusive, concurrent, repeat, out-of-sequence, parallel with synchronous, refinement and etc [14]. In reference [15] the BPEL described services are transformed into colored Petri nets, coordinators (called mediator) are introduced in model combination to realize services composition. In reference [16] the service has been transformed into a set of rules for Horn clause, user's inputs and outputs were converted to facts and goals in Horn clauses, so a problem of services combination is transformed into a Horn clause logical reasoning problem, this method is based on Zhovtobryukh's goal driven approach, but it's a detailed implementation and is more in-depth.

Generally speaking an Internetware structure should contain a fundamental component layer and management or control layers on it. As what we have discussed in section II, current ADLs either support only description and analysis of senior level abstraction of system architecture, or use different languages in criterion specification and system implementation; they are difficult to support the complete software developing process. An idea ADL should provide not only good abstract expression ability, but also analysis and reasoning ability in the stage of architecture designing. It's also a good idea to support the smooth transition from architecture description to code implementation of transition, so that a good design can deduce a good realization. Petri nets possess all the capabilities discussed above.

Software architecture is the first time map from system requirements to system design elements, plays the role of bridge between requirements and design. To define system composition and system structure using Petri nets, reflect relationship between system requirements and system structure elements, some basic problems should be solved in advance, these problems including:

(1)the research of appropriate component model structure, using a WCS oriented Petri net (WCS-net) model to describe component structure, connection between components, as well as the architecture

configuration. It's necessary to define a set of architecture levels of grammar and explain its semantics in the perspective of token, transition and arc constraint.

(2)the research of component control strategy and implementation technology in the perspective of system structure, environment interaction, dynamic strain and trusted control, so as to implement static, dynamic evolution of software structure.

(3) Provide efficient, reliable service discovery algorithms to insure the efficiency, rationality and trustworthiness in components combination. Different from ordinary Petri nets, a WCS oriented Petri net should provide the ability of transition service specification, including detailed service information of subnet (a combination of components) and the information profile of underlying component services.

Currently Petri nets based WSC researches lay stress on indirect modeling, namely the transition from models described using existing ADLs such as BPEL, to corresponding models described by Petri nets, few researches are performed in the field of direct modeling by Petri nets. For component-based software service description there are three key points: description of the features of the service, service implementation, service interface [5]. Most of the researches focus on model implementation description, while service interfaces and service features description are less involved.

The advantage of current research approach is that researchers can make full use of existing WSC ADL research results, and the dynamic modeling ability of Petri nets, with these two together to solve problems currently encountered in architecture modeling; but in the long run this indirect modeling methods will reduce the efficiency of system development and increase system development costs, because you have to transform between different models and different standards. Another problem in current research is less attention was paid to design specific service discovery and combination algorithm according to features of Petri net model.

IV. SOFTWARE ARCHITECTURE MODEL

A. Basic Principle

To build architecture model using Petri nets, components correspond to transitions (which can be refined), the relationships between components, including dependencies, invoke, collaboration, and so on, can be represented by places and arcs, component clustering is reflected by the hierarchical structure, different initial states represent the generalization, system resource and environmental constraints are represented by tokens in natural, in this way the model can be calculated easily. Based on the proposed model, the component-based software evolution, including the operation of component appending, deletion, modification, parameters adjustment and associated relationship changes are also available, also we can evaluate the result of a change or member's contributions, etc.

B. Architecture Model

As the extended SOA model in figure 1, service composition layer can be introduced, in which the components combination relationship model can be established by WSC-nets, this layer can be further decomposed into many sub layers if necessary. The components are divided into atomic component and composite component, an atomic component is the fundamental structure corresponding to very simple subnet which is unnecessary to be divided any longer, in figure 2, the atomic component includes *A, C, B2, B3, B1.1 ,B1.2, B1.3*, they are represented by dotted circles; a composite component is composed by subnets by fusion or in a hierarchical manner, in figure 2, the composite component includes *B* and *B1*, they are represented by real circles. The connections between components are represented by arcs, in such a model, the links between transition nodes can be physical, as the relationship between upstream and downstream components in the same physical system; but more commonly the nodes are logically connected, because the corresponding node is possible to be a composite component node, it may be a non atomic components, in addition the corresponding component may locate in certain other places, this is the characteristics of WSC.

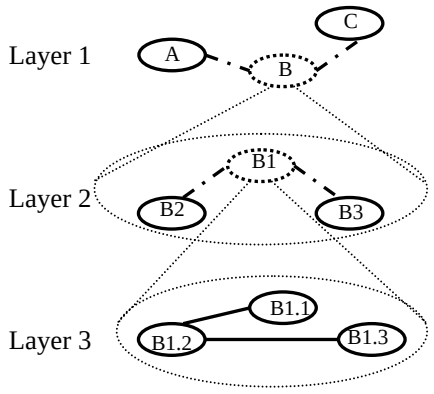

Figure 2. atomic components and composite components

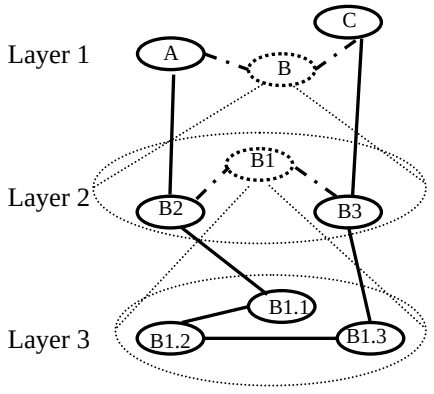

Figure 3. logical connection vs. physical connection

In figure 2, the physical connections are represented by real lines, like the connection between *B1.1* and *B1.2*; the logical connections are represented by dotted lines, like

the connection between *A*, *B* and *C*. The actual physical connection between components may need to go through the model layer by layer and be implemented down to a lower level like figure 3, here the logical connection between *A* and *C* in figure 1 is realized by the physical connections of *A - B2 - B1.1 - B1.2 - B1.3 - B3 – C*.

By establishing a layered service composition model (corresponds to scheduling layer) with WSC-nets, a task is decomposed layer-by-layer like figure 2; the Petri net model forms a natural relationship diagram at different levels. The service composition process is converted to go through the graphs to find paths from input to output or from output to input. Path discovery can be based on an existing graph searching algorithm, including forward searching or backtracking algorithm, they should be redesigned or improved to make it suitable for WSC-nets models. Thanks to the hierarchical model, the searching operation is executed layer by layer and each search is limited to the same logical hierarchy, so you can avoid the traditional state space explosion problem when graph searching algorithm are used in a complex model.

In a WSC-nets model, a regional locally centralized control policy is adapted, each composite node manages the service information and knowledge of its subnet nodes, but in the same layer it is a distributed control strategy, providing accurate information of components. The connected arcs in a WSC-net model are divided into logical connection arc and physically connected arc as we have discussed above, typically the descriptions of component library structure in a WSC-nets model consists of logical connections, physical connections between nodes are established only when members of combination service are decided finally, the physical connected parts constitute a subset of the original model.

C. Architecture Evolution

The decision controller is the core part of a system; it is designed not only to meet the needs of users' different targets, but also to select a correct treatment plan in case of a system error, or the re-composition of components. At the same time the decision controller is responsible for assembling components in the same node, deciding synchronization and collaboration mechanisms among different sites. In a Petri net model, the change of the system structure, or the relationship adjustment among components, is represented as subnets fusion, or re-fusion. According to the features of applications, the main operations are fusions of places; the fusion of transitions takes place mainly on the occasion of function-oriented reorganization.

Target changes of upper layers will result in the policy of change management layer to be added, deleted or modified. Kramer suggested a number of requirements on how to describe targets in his model, but gave no answer for the problems occurred to meet these requirements. In a Petri net model, system targets are represented by different tokens, so modification of a target is actually the process to assign a different initial state for the model, if the given tokens are different, different paths are selected, this in fact indicates different evolutionary scenario. Past research of Petri nets based knowledge expression helps to solve this problem, a control strategy is represented as a rule, and the evolution process can be realized through forward and reverse deduction of rules.

D. Reduction Rules Design

The design of reduction rule set has to achieve a reasonable balance between model transformation capabilities (the completeness) and usability in application; current SISO subnet based transformation rules restrict the application greatly because of too strict restrictions. Abstraction and refinement are inverted operations of reduction rules in two opposite directions to expand or compress net model respectively. Existing research results of basic reduction rules, such as self circulation elimination, continuous place or transition elimination, equivalent place or transition elimination, can be viewed as the reduction of simple subnets, the design of the reduction rules set is actually the expansion and extension of existing rules, to find out restriction rules for the more larger size subnets. Our previous studies show that some MIMO subnets can be transformed into a transition or a place; if certain conditions are met, some MIMO subnets can be translated into corresponding SISO subnets, or solve such problems by coloring.

V. SUBNET BASED WEB SERVICES COMPOSITION

Components are reusable software elements that can be used to construct different software system. The open software production based on components came up into being because of the needs from large-scale industrial software production. The software development process is divided into two stages: reusable component production and component composition, details to solve individual problem in program design is shielded gradually. The main task of component production industry is to produce a large amount of reusable components to solve different problems; Component assembly industry's task is to produce required software systems based on those mature components available. A component composes of two parts: component interface and component body, component body implements various features of the component, it can be realized by popular technologies today; the component interface supports component assembly. The concept "*component*" is divided into component class and component instance, a instance is generated by assigning parameters for corresponding component class, appropriate application software are constructed through instance assembly and control.

In a Petri net model, a component corresponds to a subnet. Although Petri nets have become an important modeling method today and are widely used, modeling complex systems with common Petri nets (for example, the P/T net) directly in a large and complex system will certainly result in model state space explosion. To overcome this problem, the first method is to simplify model through transformations under particular application environment, or other transformations which

can maintain certain important properties, so as to reduce the state space and simplify the analysis process. Hierarchy and modularization technology is the other effective method, through abstraction step by step and simulation block by block, the complexity of system design can be greatly reduced, this is the core idea in structural design. From the perspective of software engineering, the hierarchy and modularization method is much more promising; it can popularize Petri nets to more wide area and high level application, this method lies in subnet technology.

The proposed subnet structure should meet the requirement of hierarchical modeling in Internetwares, the component subnet based WCS-net is defined on that. The research of WCS-net can base on the existing subnet research, combined with the concepts of interface standardization and subnet normalization proposed by us in reference[21], taking subnet's external environment and basic internal structure and design requirements into account at the same time. The component subnet should meet the following conditions:

(1) The restrictions for object system are relatively relaxed compared with existing research results so as to increase modeling flexibility;

(2) Component subnet based operations, including abstract, refinement and the fusion between subnets, maintain the necessary properties of the model so that the reliability and trustiness of service composition are ensured.

Under reasonably conditions, transformation on component based subnet should guarantee the equivalence between subnet and corresponding transition, that is, the model preserves original important properties such as boundedness, safeness, deadlock-free, etc. before and after its model transformation. Criterions for component subnet should be decided elaborately according to appropriate subnet characteristics; the non-standard subnet should be transformed into standard subnet using specific methods if possible. It's also important to define operations including component subnet abstraction and transition refinement, decide basic rules for WCS-net based direct system modeling method.

Existing research results usually have a more demanding requirements, that is, the system should be fully recovered after a complete routine running. A reasonable assumption is that the system returns to the original state or a state equivalent to that in some form after a loop runs, namely it only requires that the system has to be recoverable; in fact you can find even the recovery is not required for many systems, there exists too many normal system who has some irreversible states. So the more fundamental requirement is: in an external environment meeting the requirements, the system can always run successfully, this indicates the independence of modules. Simply put, in premise of trustiness, subnet needs to have the ability to run continuously, if a past running will result in subnet state changes and such changes make the system lose its processing power, then the design is unsuccessful.

A. Ensuring Subnet's Running Ability

Subnet's persisting running ability essentially requires subnets and transitions output in the same way if their inputs are same; it indicates the equivalence between subnet and transition in the process of abstract or refinement, it also indicates that the subnet can be trusted. The first step is to do is to make sure that the interface and service is consistent before and after transition/subnet substitution, namely the "likeness", realized by interface standardization and subnet normalization, this is especially important for relatively complex interfaces or external environment. Compared with subnet abstraction, transition refinement is more complex because there usually exist different refining scenarios, so there are a number of possible choices and this brings uncertainty, it is critical to make reasonable refinement principle.

"Similar behavior" is based on "Likeness", which aims to keep as much properties of the original model as possible for the transformed model, such as boundedness, aliveness, etc. Nevertheless, because the subnet is fired step by step, it is impossible to achieve absolute equivalence between transition and subnet, so the question is, to what extent and how to realize the relative equivalence between the two. The difference between subnet and transition is caused by the "half-triggered" state, by "half-triggered" state analysis and loopback testing analysis, we can solve the problem of "similar behavior". Take practical application, for example production route analysis into account, the resulted Internetware oriented subnet structure — the component subnet can be obtained.

Aliveness of subnet does not guarantee that the subnet can run continually because the refined model (or the model before abstraction) is more complex, the models may enter the multiple-time-fired but uncompleted state relative to the pre-refinement (or post-abstraction) state, this brings resource competition and lead to certain uncompleted firing sequence. Therefore, although the component subnet does not emphasize recoverability, it must be deadlock free.

B. Component Complexity Analysis and Control

The reason to research component based subnet structure is to reduce the complexity of the Petri net model and avoid state space explosion. Reasonable modularization is based on careful analysis of business process, it is necessary to ensure the module with reasonable granularity, meanwhile the complexity of subnet is reasonably too, complexity evaluation methods such as Halstead and graph analysis method of Petri nets can be used here to assess the complexity of modules. For the development of new component, the basic module division principles of high cohesion and low coupling should be insisted, and to put forward suitable subsystem partitioning algorithm to decide the range of subnet.

Clustering is a possible way to solve the problem of automatic subnet division; the algorithm should be designed according to the known principles of subnet division. Combined with the module complexity assessment results, weight of different parameters can be

adjusted according to system running results. For existed components in the lower level, the upper layer model can use methods including target decomposition and service matching, and take factors such as efficiency, complexity into account, so as to reach a balance between fan and depth. A trusted complex software model can be established by elaborate design of basic structure, behavior pattern, run mechanism, evolution rule and trusted guarantee mechanism.

VI. Develop Internetware Using Petri Nets as Unified model

Through our previous studies we find Petri nets works well in different system layers, take the A^3 model for Enterprise-Informatization System[17] as an example, Petri nets are widely used from automatically control to operation management and enterprise decision, in every different level there exists successful applications. To solve a problem the realization of corresponding system includes some or all of the processes from requirements analysis, system design, system implementation, system analysis to system testing. But generally speaking, the characteristic of current Petri net applications is that they are widely used but focused on the lower layers application of the A^3 model and there lack a unified standard.

On the basis of existing related research and application, we can present a generic method for Petri nets based systems analysis and design. Analysis and design specifications should be decided by integrating both direct and indirect modeling methods, the methods we raised in our previous studies to analyze module feature, data completeness and consistency based on the associated matrix forms the basis to do so[18,19].

It's true that the lifecycle of Internetwares differs from the traditional concept of life-cycle, for Internetwares it takes on a "big life-cycle" concept [20], but the applications of Petri nets in previous research still contribute to the development of Internetwares, the unified Petri net model in all system stages makes it more friendly and efficient to realize the repetitious of various stages in the "big life-cycle" development process.

VII. Conclusion

In above sections we discuss the idea of how to solve problems met in Internetware development today by Petri net method. Although people mainly regard Petri nets as a small scale tool for modeling, simulation and analysis of specific problems, we believe this view can be changed in a long term, Petri nets can be used as a systematic engineering method for system development. System analysis and design based on the same model can solve the problems of disjointedness between different stages, through the research of Petri nets based system modeling and analysis methods we can realize the long term target of natural transition among different stages of Internetware developing, including system analysis, design, implementation and running. Coexistence of different technologies, different methods and different models in the same system is the main obstacle to realize system integration, a unified Petri nets based model and tool provide a key to overcome this problem. The idea of "same system uniform model, same model multiple functions" will improve system efficiency and reliability greatly. Taking full advantage of the Petri nets, that's the combination of straightforward presentation skills and the mathematical analysis ability, provides an effective method for Internetwares development.

Acknowledgment

Project supported by the Natural Science Foundation of Guangdong Province (Grant No. 8451032001001610), Natural Science Foundation of Guangdong Province (Grant No. 0630970).

References

[1] L.C.Tian, L.Zhang, B.S.Zhou, "Analysis of the Current Status of Architecture Description Languages", Computer Science, 2005, vol 32(2), pp.109-113.

[2] G. Booch, I.Jacobson, J.Rumbaugh, "Unified Modeling Language", version 1.0, Rational Software Corporation, 1997.

[3] L.J.Osterweil "Formalisms to Support the Definition of Processes", Journal of Computer Science and Technology 24(2): 198 – 211 Mar. 2009

[4] X.Y.Zhu, Z.S.Tang, "A Temporal Logic-Based Software Architecture Description Language XYZ / ADL", Journal of Softwar,2003,vol. l4(4),pp.713~72.

[5] M.ter.Beek, A.Bucchiarone, S.Gnesi, "A Survey on Service Composition Approaches: From Industrial Standards to Formal Methods", Technical report, 2006.

[6] M.P.Papazoglou, D.Georgakopoulos, "Service-oriented computing", Communications of the ACM, 2003, vol.46(10),pp.25–28.

[7] J.Kramer, J.Magee, "A rigorous architectural approach to adaptive software engineering", Journal of Computer Science and Technology,2009,vol.24(2),pp.183 – 188.

[8] H. M. Wang, Y. B. Tang, G.Yin, "Trustworthiness of Internet-based Software", Science in China (Series F), 2006, vol.49(6),pp.755-773.

[9] Bazil Parv, Simona Motogna, "Comdevalco- A Framework for Software Component Definition, Validation, and Composition", Studia Univ. Babes-Bolyai, 2007,LII(2), pp.59-68.

[10] J.Lv, X.X.Ma, X.P.Tao, "Research and development of Internetware", Science in China (Series E) , 2006, vol.36(10), pp.1037-1080.

[11] W.M.P. van der Aalst, K.Bisgaard Lassen, "Translating Unstructured Workflow Processes to Readable BPEL: Theory and Implementation", Information and Software Technology, 2008, vol.50(3), pp.131-159.

[12] Niels Lohmann, Peter Massuthe, Christian Stahl, and Daniela Weinberg, "Analyzing Interacting WS-BPEL Processes Using Exible Model Generation", Data Knowl. Eng., 2008,vol.64(1),pp.38-54.

[13] R. Hamadi and B. Benatallah, "A Petri Net-based Model for Web Service Composition", 14th Australian Database Conference (ADC 2003), Adelaide, South Australia, 2003.

[14] Dmytro Zhovtobryukh, "A Petri Net-based Approach for Automated Goal-Driven Web Service Composition", Simulation, 2007, vol.83(1),pp.33–63.

[15] W.Tan, Y.S.Fan, M.C.Zhou, "A Petri net-based method for compatibility analysis and composition of Web services in

business process execution language", IEEE Transactions on Automation Science and Engineering, 2009, vol.6(1),pp. 94~106.
[16] X.F.Tang, C.J.Jiang, Z.J.Ding,C.Wang, "A Petri Net-Based Semantic Web Service Automatic Composition Method", Journal of Software,2007,vol.18(12),pp.2291-3000.
[17] Z.J.Wang, Z.X.Cai, "A Petri Net Based Unified Modeling Method for Enterprise-Informatization System", Control Conference(CCC 2007), Zhangjiajie, China, pp.791-795.
[18] Z.J.Wang, Z.X.Cai, "Indirect Modeling Method by Translating IDEF0 Model into Petri Net Model", Journal of System Simulation, 2008,vol.20（15）,pp.3915-3919.
[19] Z.J.Wang, Z.X.Cai, "Function/Data Analysis Method Based on Petri Net", Computer Integrated Manufacturing Systems, 2008,vol.14（6）,pp: 1194-1199.
[20] F.Q.Yang, "Thinking on the Development of Software Engineering Technology", Journal of Software, 2005, vol.16(1),pp.1-7.
[21] Z.J.Wang, D.G.Wei, "Modeling Complex System Using T-subnet Based Hierarchical Petri Nets", Journal of Computers, 2009, vol. 4(9), pp.829-836.

Zhijian Wang was born in Changsha, Hunan Province, China in 1970. He obtained his Ph.D. degree in Control Theory and Control Engineering from Central South University in 2007 in China.

He is presently a professor of Computer Science in Information Science School, Guangdong University of Business Studies, Guangzhou, China. He is also the senior member of China Computer Federation. His current research interests include Petri nets, software engineering, and Computer Integrated Manufacturing Systems.

Yuping Hu was born in 1969, he received his B.S. degree in celestial survey from Chinese Academy of Science, China , in 1996 and his Ph.d. degree in computer science from Huazhong University of Science and Technology, Wuhan , China in 2005.He is currently pursuing the postdoctoral research in computer applications from Central South University , Changsha , China .

Dr.Hu is a professor in the Guangdong Province Key Lab of EC Market Application Technology, Guangdong University of Business Studies, Guangzhou, China. His current research interests include digital watermarking, image processing,multimedia and network security.

Shaohua Li was born in Ganzhou, Jiangxi Province, China in 1964. He obtained his M.S. degree in Computer Graphics from Beijing University of Aeronautics & Astronautics in 1987 in China.

He is presently a associate professor of Computer Science in Information Science School, Guangdong University of Business Studies, Guangzhou, China. He is also the senior member of China Computer Federation. His current research interests include Parameterized Complexity and algorithm design & analysis.

Dingguo Wei was born in Hunan Province, China in 1964. He obtained his Ph.D. degree in Computer Software and Theory from Fudan University in 2003 in China.

He is presently a professor of Computer Science in Information Science School, Guangdong University of Business Studies, Guangzhou, China. He is also the senior member of China Computer Federation. His current research interests include Petri nets, software engineering, and Electrical business.